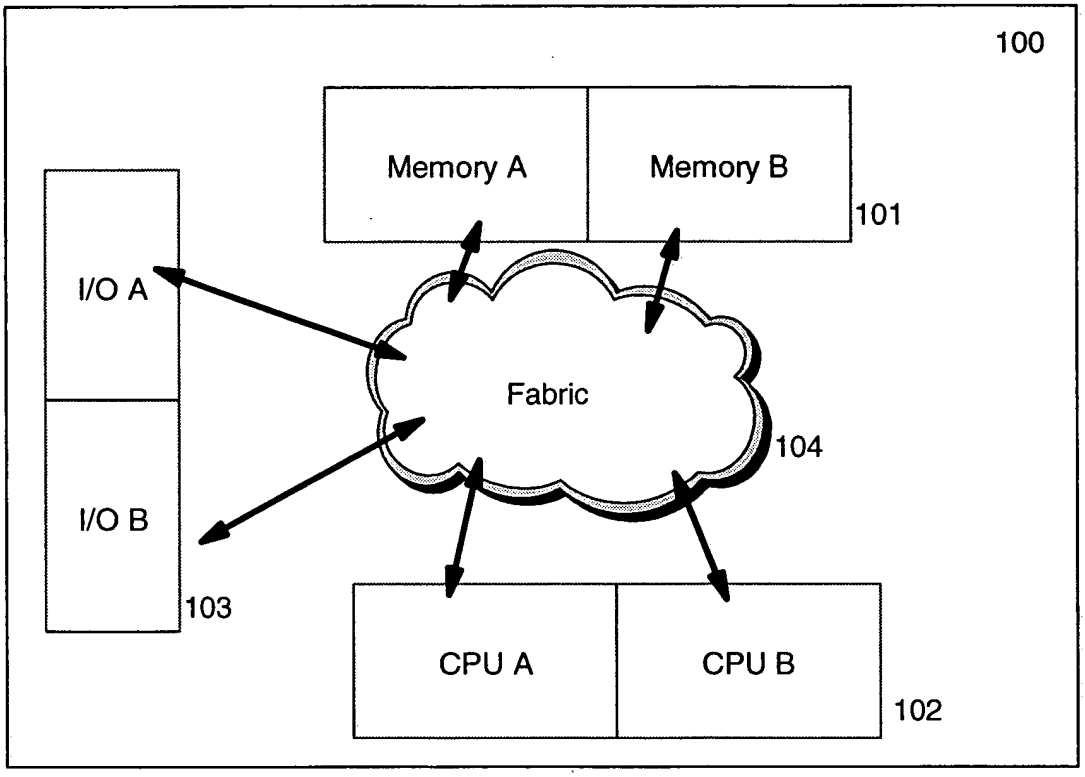



Fig 1

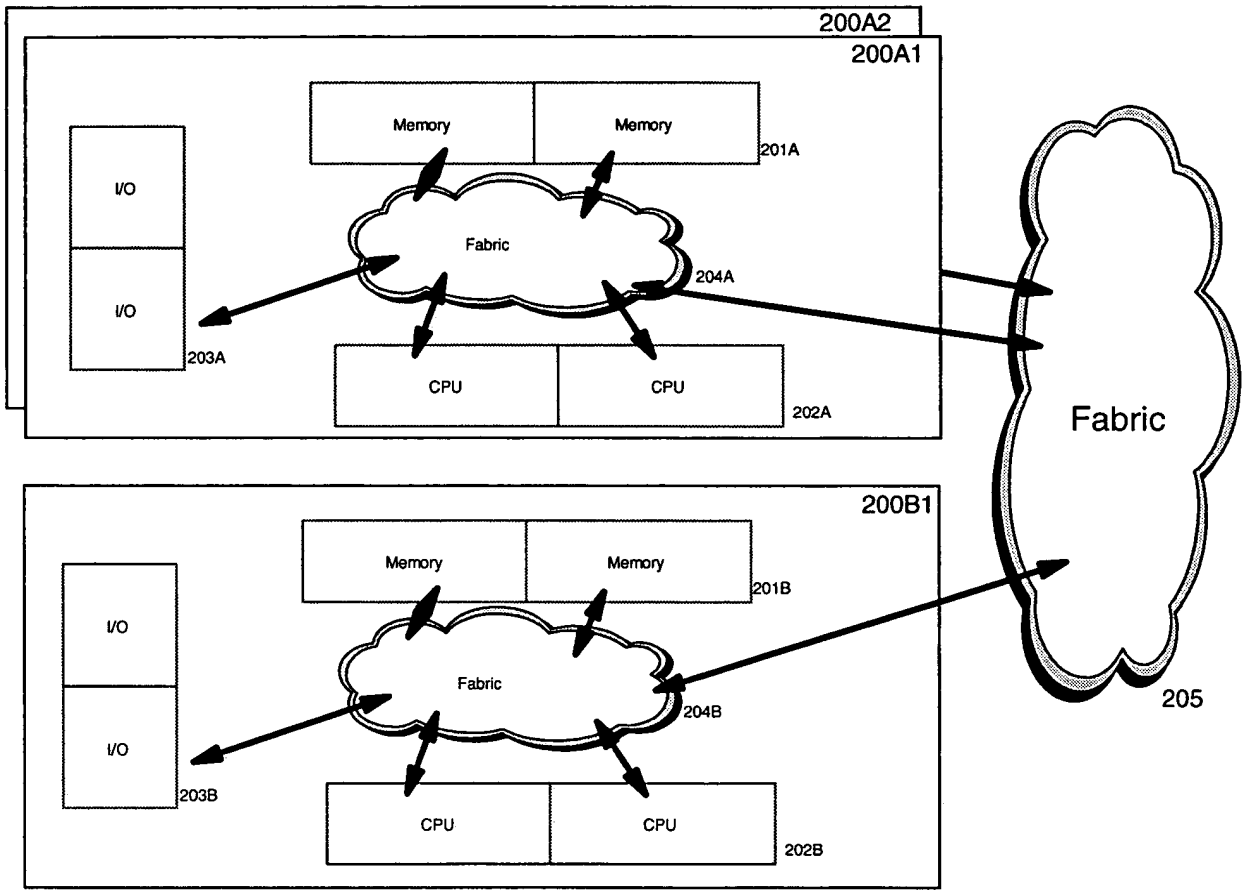

200A2
200A1
Memory
Memory
201A
I/O
I/O
203A
Fabric
204A
CPU
CPU
202A
Fabric
200B1
Memory
Memory
201B
I/O
I/O
203B
Fabric
204B
CPU
CPU
202B
205

Fig 2

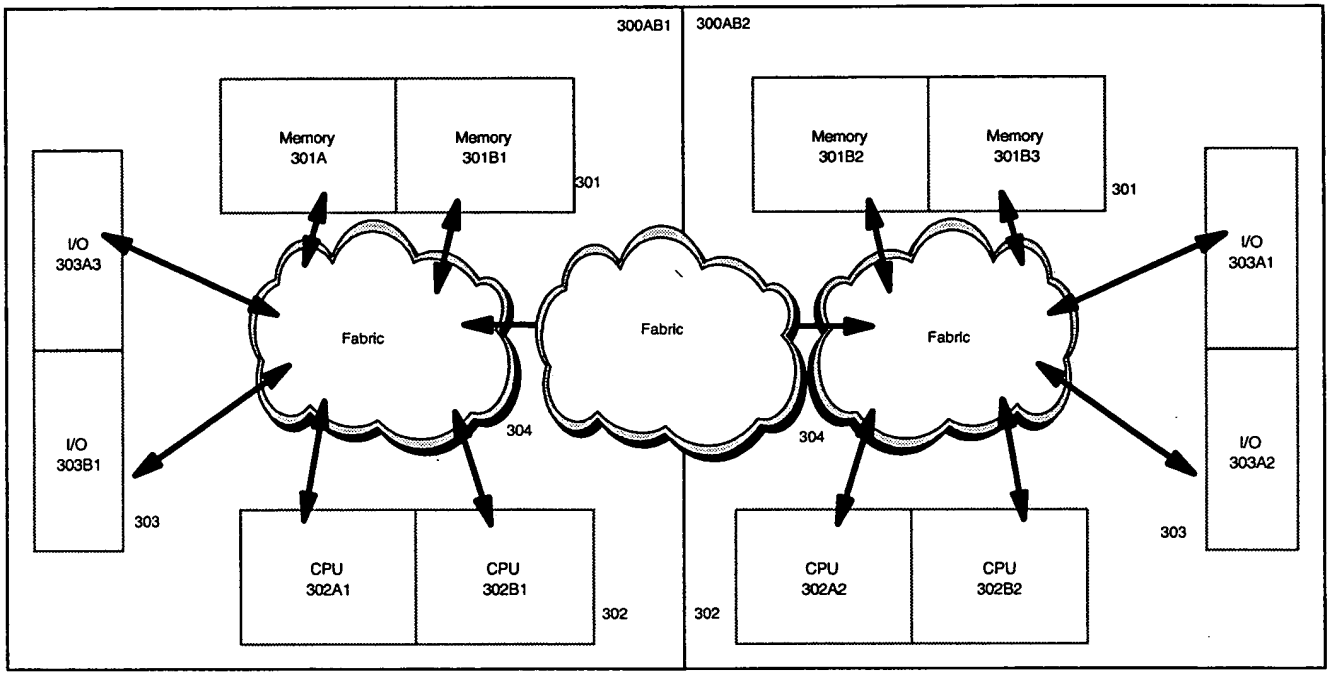

300AB1
300AB2
Memory
301A
Memory
301B1
301
Memory
301B2
Memory
301B3
301
I/O
303A3
I/O
303B1
303
Fabric
Fabric
Fabric
304
304
I/O
303A1
I/O
303A2
303
CPU
302A1
CPU
302B1
302
302
CPU
302A2
CPU
302B2

Fig 3

Virtualization allows sharing of CPUs and I/O elements by multiple partitions

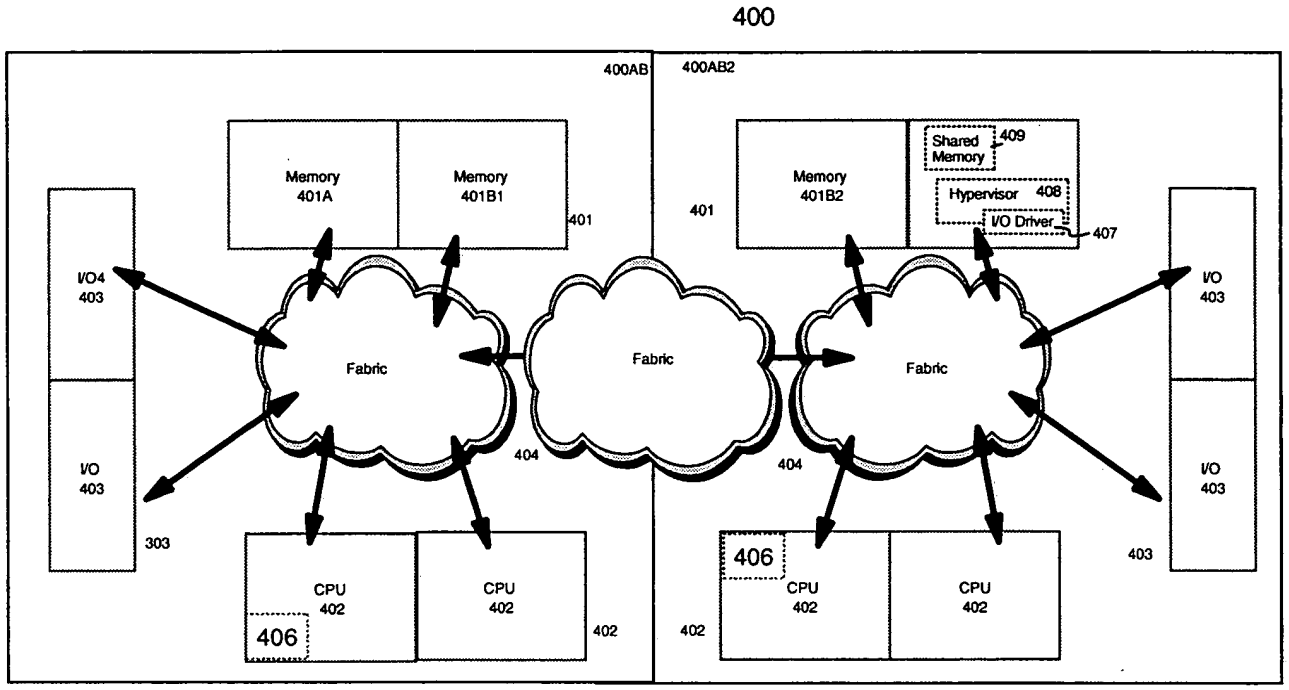



Fig 4

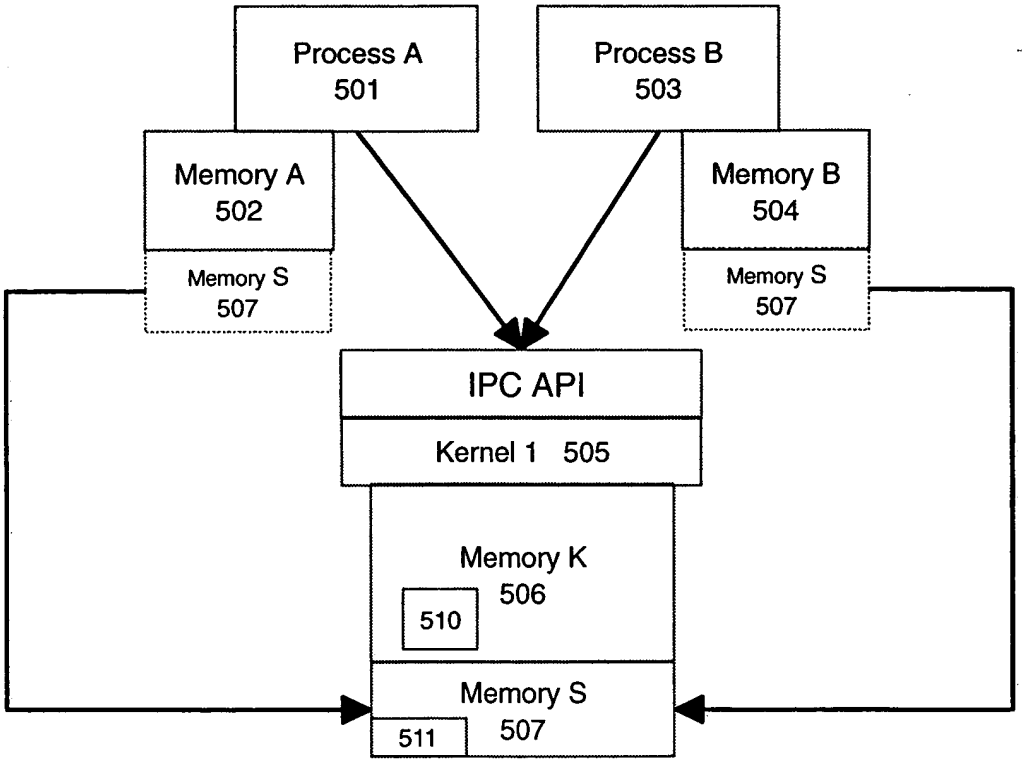

Process A
501
Process B
503
Memory A
502
Memory B
504
Memory S
507
Memory S
507
IPC API
Kernel 1 505
Memory K
506
510
Memory S
507
511

Fig 5

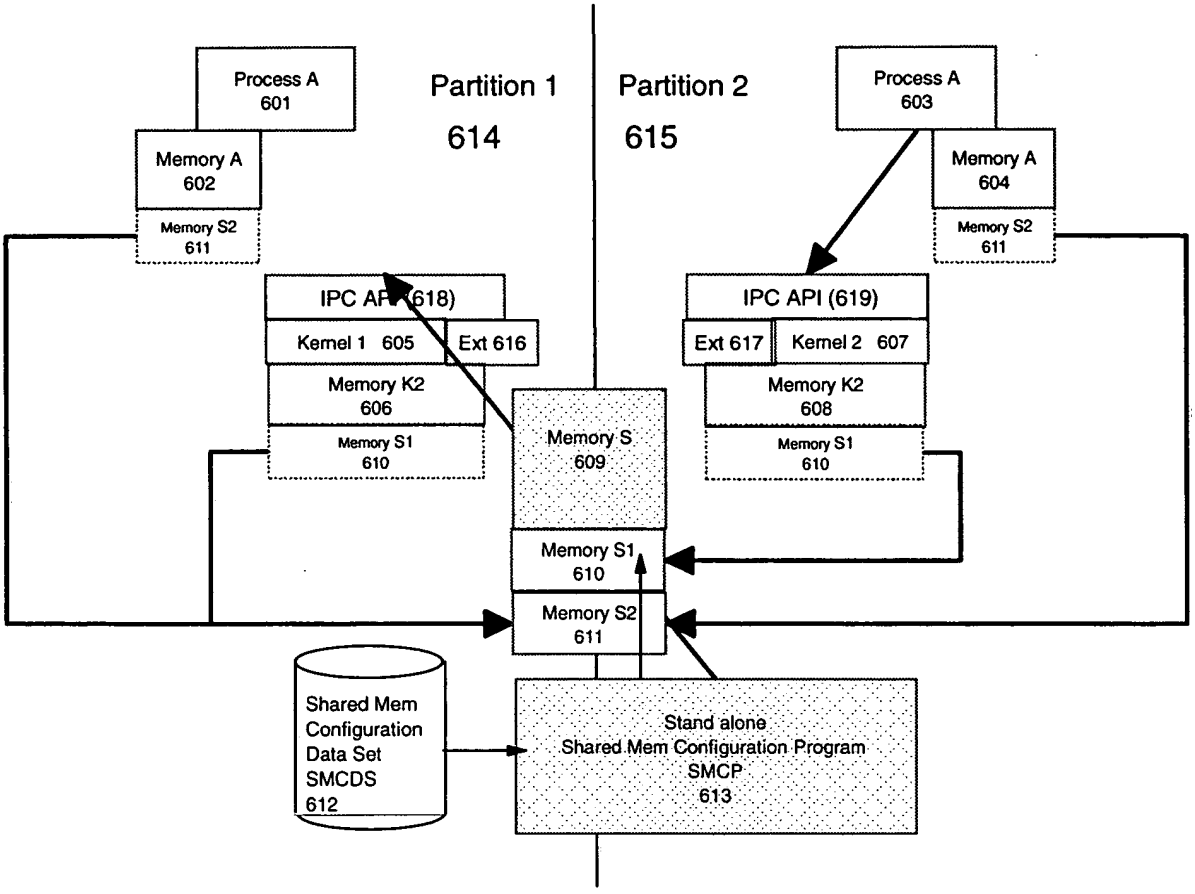

Process A
601
Partition 1
614
Partition 2
615
Process A
603
Memory A
602
Memory S2
611
Memory A
604
Memory S2
611
IPC API (618)
Kernel 1 605
Ext 616
IPC API (619)
Ext 617
Kernel 2 607
Memory K2
606
Memory S1
610
Memory S
609
Memory K2
608
Memory S1
610
Memory S1
610
Memory S2
611
Shared Mem
Configuration
Data Set
SMCDS
612
Stand alone
Shared Mem Configuration Program
SMCP
613

Fig 6

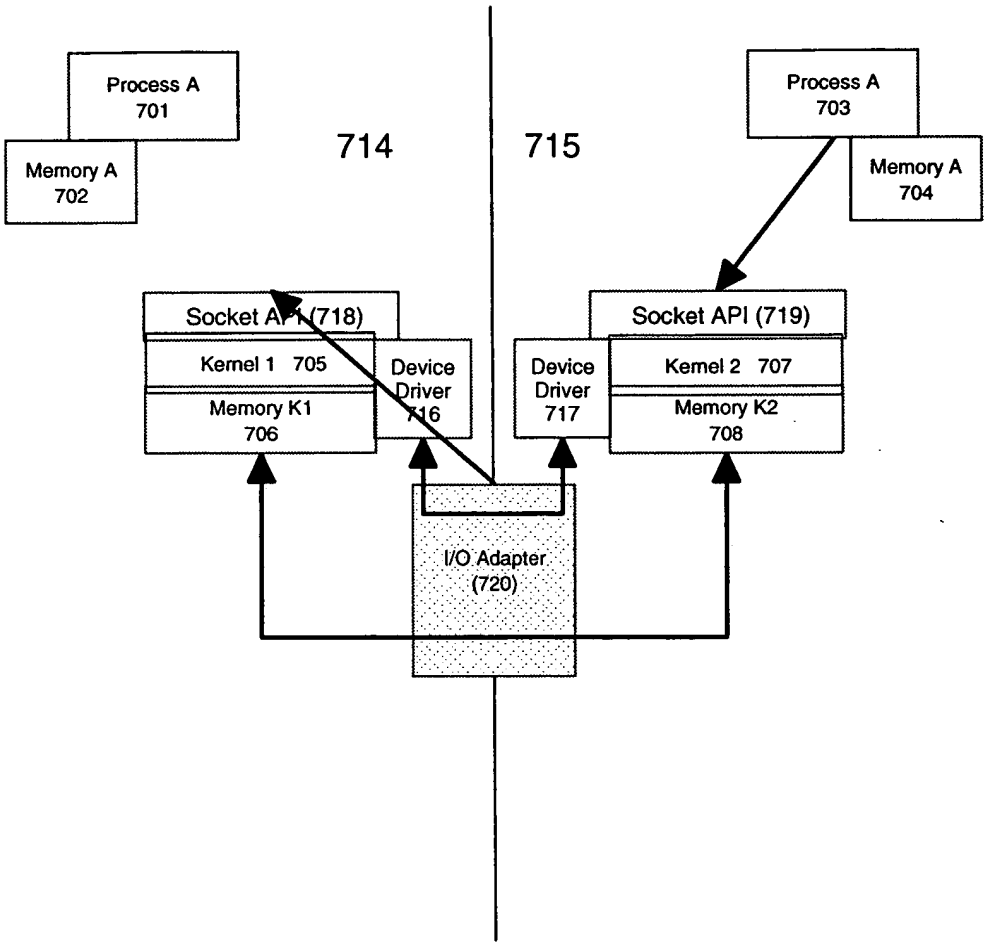
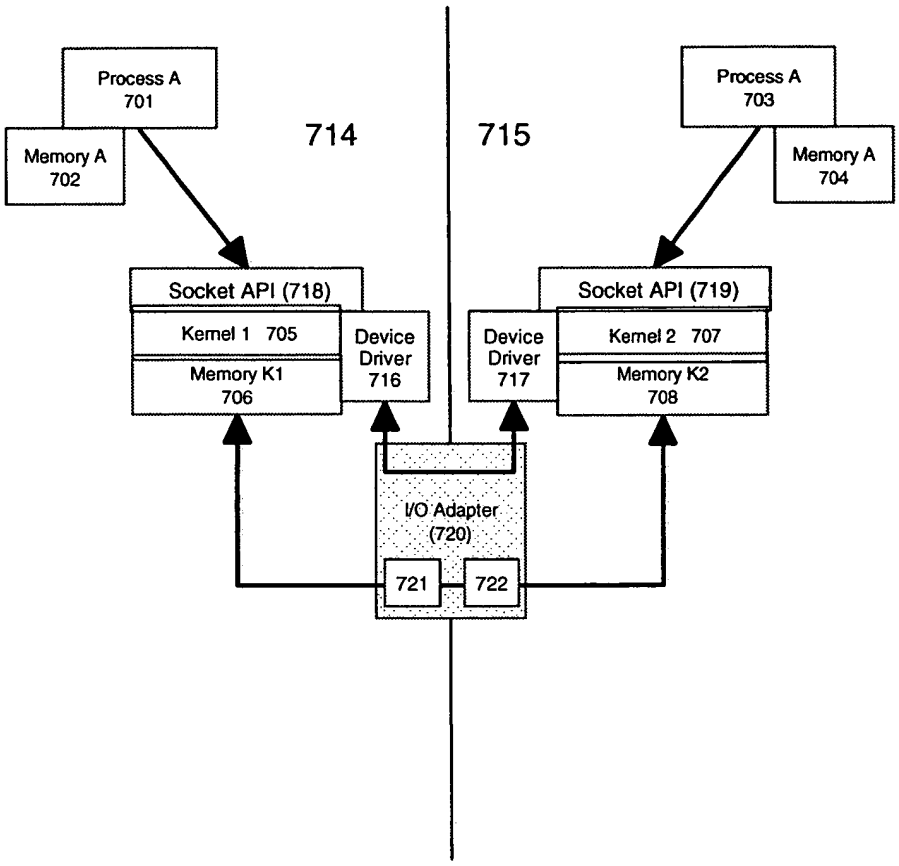

Process A
701
Memory A
702
714
715
Process A
703
Memory A
704
Socket API (718)
Kernel 1 705
Memory K1
706
Device
Driver
716
Device
Driver
717
Socket API (719)
Kernel 2 707
Memory K2
708
I/O Adapter
(720)

Fig 7A




**The Prior Art**
**Fig 7B**

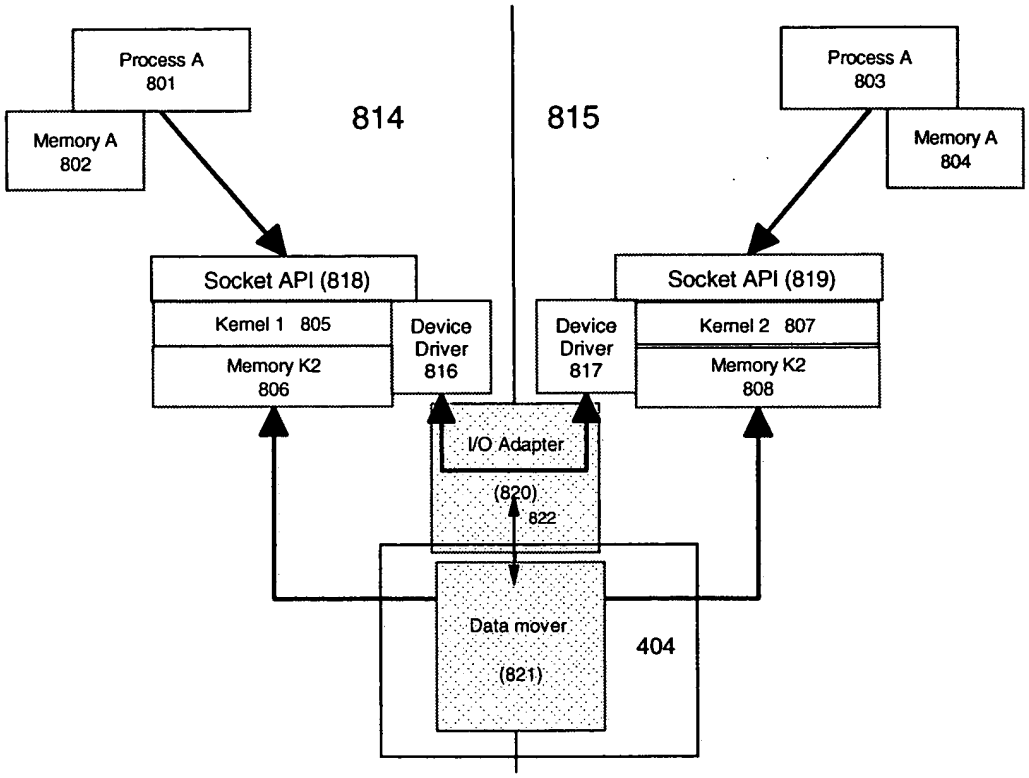
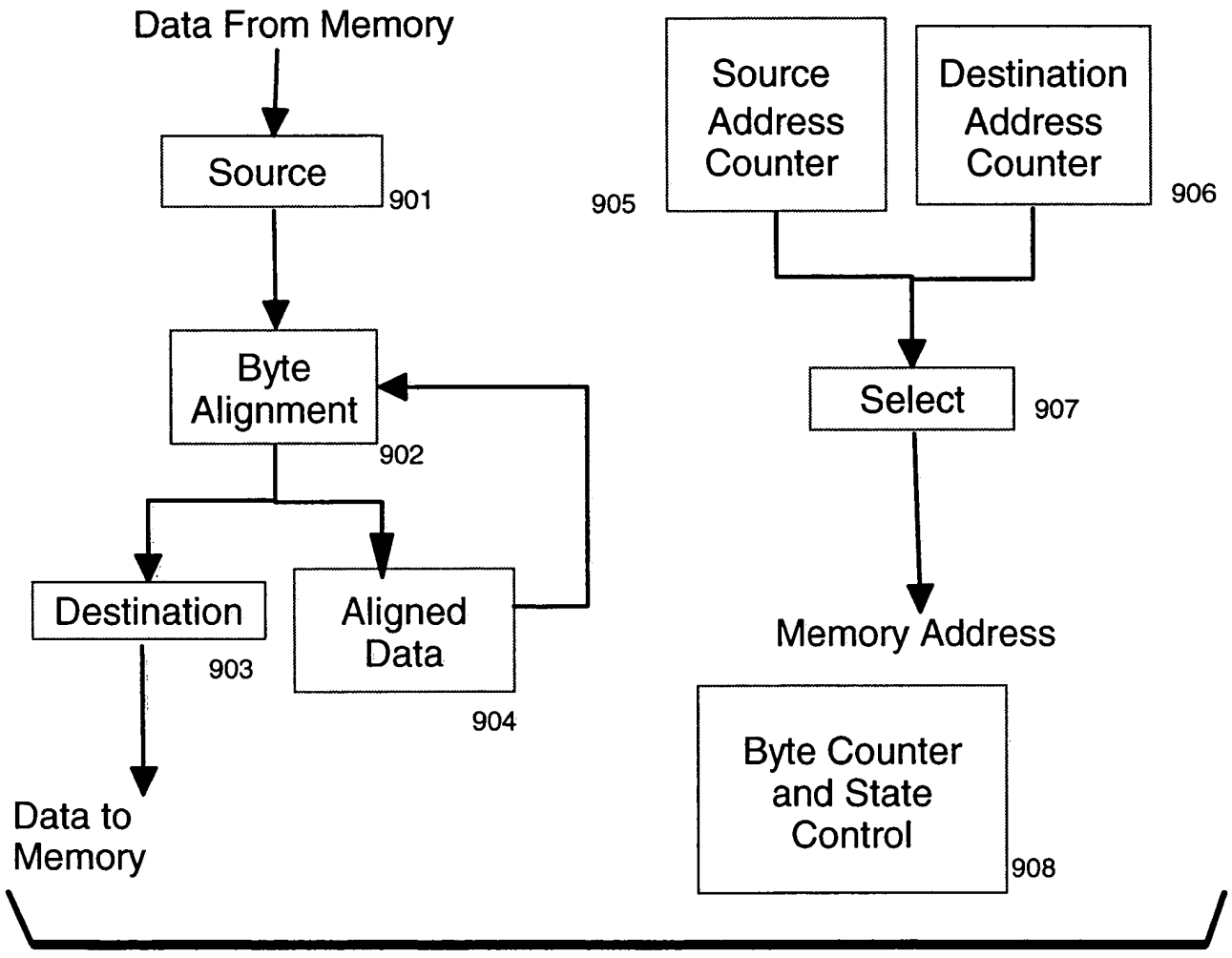

Process A
801
Memory A
802
814
815
Process A
803
Memory A
804
Socket API (818)
Kernel 1 805
Memory K2
806
Device
Driver
816
Device
Driver
817
Socket API (819)
Kernel 2 807
Memory K2
808
I/O Adapter
(820)
822
Data mover
(821)
404

Fig 8

# Fig 9


Data From Memory
Source
901
Byte Alignment
902
Destination
903
Aligned Data
904
Data to Memory
905
Source Address Counter
Destination Address Counter
906
Select
907
Memory Address
Byte Counter and State Control
908

# Fig 10

1000

| MVXL | Count | Destination | Source |
|---|---|---|---|

MVXL moves the number of bytes specified by the count register from the physical address specified by the source register to the physical address specified by the destination register. The instruction is priveleged.

(MVCL performs the same function between virtual addressses.) Here the Device Driver loads the register with physical rather than virtual addresses allowing cross partition data movement.

# Fig 11

1101. User calls Device Driver
   - Supplies
     Source Network ID
     Source Offset
     Destination Network ID

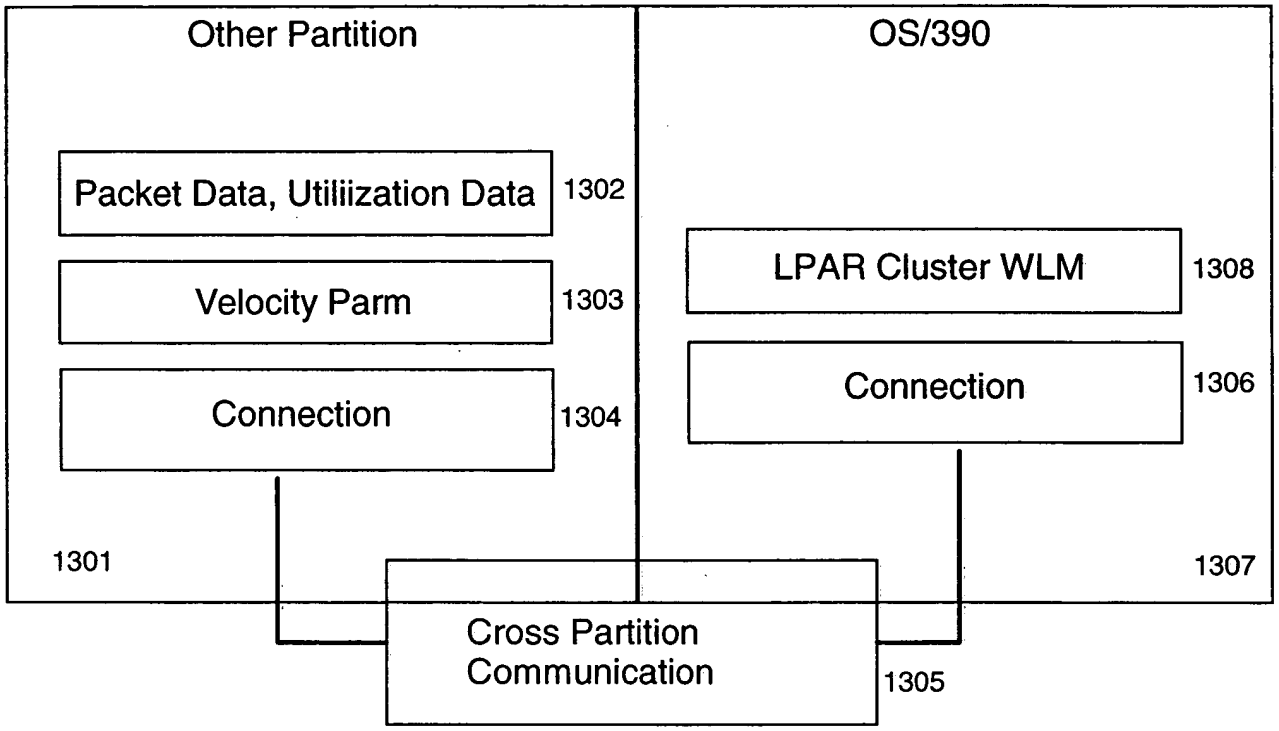
1102. Device driver transfers addresses to Adapter
1103. Adapter Translates Addresses
   - Looks up Phyical Base addresses from ID's (Table Lookup)
   - Obtains Lock and current Destination Offset
   - Adds offsets
   - Checks bounds

1104. Adapter loads count and addresses in registers
1105. Adapter executes Data Move
1106. Adapter Frees Lock
1107. Adapter notifies device Driver which "Returns" to user

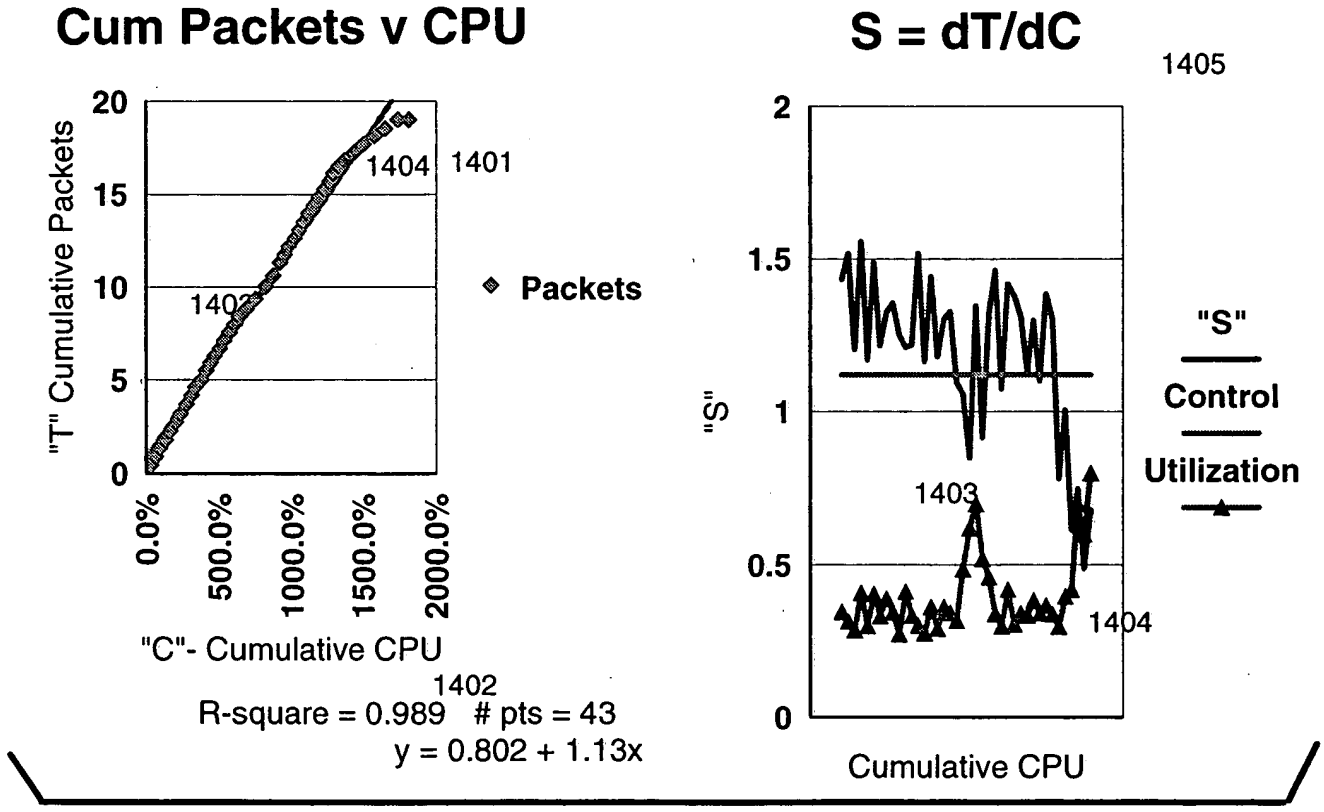
# Fig 12

1201. User calls Device Driver
    - Supplies
      Source Network ID
      Source Offset
      Destination Network ID
1202. Device driver sends addresses to adapter
1203. Adapter Translates
    - Looks up Phyical Base addresses from ID's (Table Lookup)
    - Obtains Lock and current Destination Offset*
    - Adds offsets
    - Checks bounds
    - Returns Lock and Physical addresses to Device Driver
1204. Device Driver executes Data Move
1205. Device Driver Frees Lock
1206. Device Driver Returns

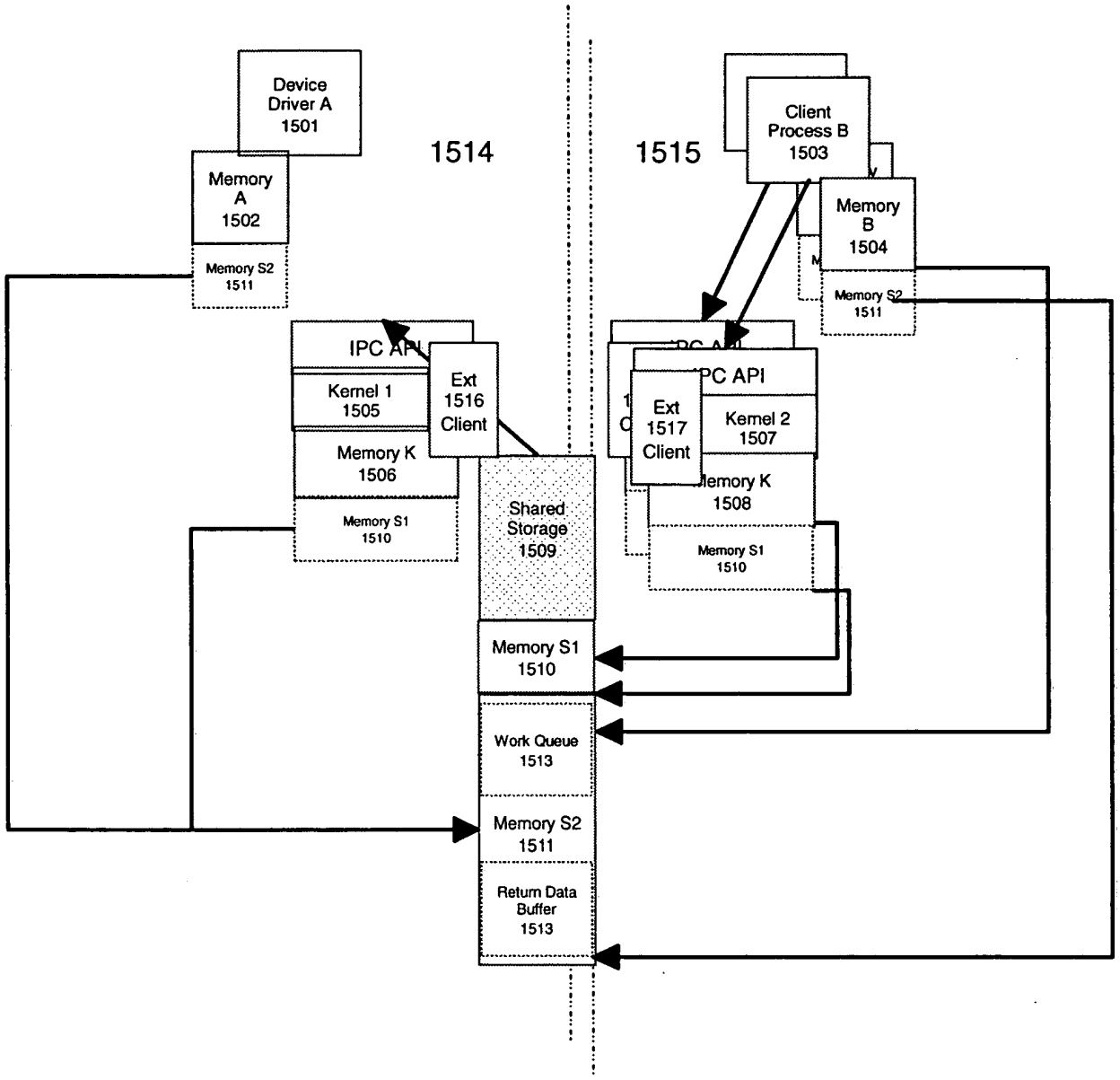
# Fig 13


Other Partition
OS/390
Packet Data, Utiliization Data
1302
Velocity Parm
1303
Connection
1304
LPAR Cluster WLM
1308
Connection
1306
1301
1307
Cross Partition
Communication
1305

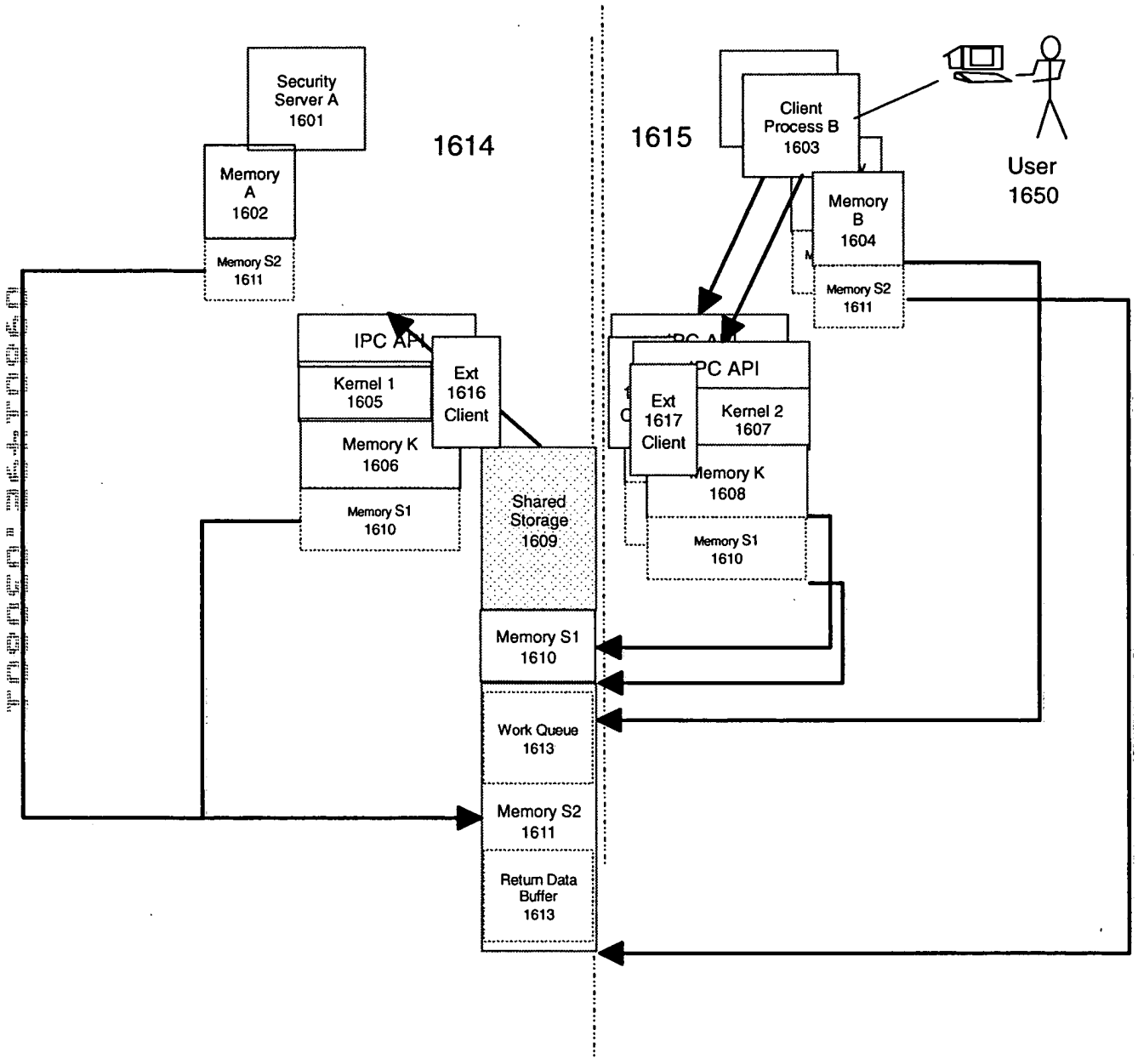
# Fig 14


Cum Packets v CPU
"T" Cumulative Packets
20
15
10
5
0
0.0%
500.0%
1000.0%
1500.0%
2000.0%
1404
1401
1403
Packets
"C"- Cumulative CPU
1402
R-square = 0.989 # pts = 43
y = 0.802 + 1.13x
S = dT/dC
1405
"S"
2
1.5
1
0.5
0
1403
1404
"S"
Control
Utilization
Cumulative CPU

# Fig 15


Device Driver A 1501
Memory A 1502
Memory S2 1511
1514
1515
Client Process B 1503
Memory B 1504
Memory S2 1511
IPC API
Kernel 1 1505
Memory K 1506
Memory S1 1510
Ext 1516 Client
Shared Storage 1509
Ext 1517 Client
Kernel 2 1507
Memory K 1508
Memory S1 1510
Memory S1 1510
Work Queue 1513
Memory S2 1511
Return Data Buffer 1513

# Fig 16


Security Server A 1601
1614
1615
Client Process B 1603
User 1650
Memory A 1602
Memory S2 1611
Memory B 1604
Memory S2 1611
IPC API
Kernel 1 1605
Ext 1616 Client
Memory K 1606
Memory S1 1610
Shared Storage 1609
IPC API
Ext 1617 Client
Kernel 2 1607
Memory K 1608
Memory S1 1610
Memory S1 1610
Work Queue 1613
Memory S2 1611
Return Data Buffer 1613